VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individální odborné praxe

Individual Professional practice in the Company

2018 Dominik Sobek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Dominik Sobek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: ATACO spol. s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Mgr. Jiří Dvorský, Ph.D.**

Konzultant bakalářské práce: Ing. Rostislav Bosák

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2018 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 30. dubna 2018 .....

Úvodem bych velice rád poděkoval všem, kteří mi při vykonávání bakalářské praxe pomáhali, zejména panu ing. Rostislavovi Bosákovi, jež mi pomohl zvyknout si na firemní prostředí a naučil mě v něm pracovat. Dále chci poděkovat vedení Vysoké školy báňské – Technické univerzity Ostrava za umožnění vykonávat praxi jako alternativu k tradiční bakalářské práci.

Abstrakt

Bakalářskou praxi jsem prováděl ve firmě ATACO spol. s r.o. Pracoval jsem na webové aplikaci pro zobrazení a správu prodlev vznikajících ve Středojemné válcovně, což je závod firmy ArcelorMittal Ostrava a.s. Mým úkolem tedy bylo vytvoření aplikace, se kterou budou schopni pracovat dělníci na provoze, a to tak, aby bylo ovládání intuitivní a rychlé.

Klíčová slova: Hibernate, Vaadin, Java, JUnit, KeyCloak, WildFly

Abstract

I did my bachelor's practice at ATACO spol. s r.o. I worked on a web application for viewing and managing the delays that occur in the Rolling Mill, which is the plant of ArcelorMittal Ostrava a.s. My job was to create an application with a simple and intuitive interface the workers would be able to work with.

Key Words: Hibernate, Vaadin, Java, JUnit, KeyCloak, WildFly

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
BIC	–	Business innovation centre Ostrava
CSS	–	Cascading Style Sheets
DM	–	Doménový model
EJB	–	Enterprise Java Beans
FEI	–	Fakulta elektroniky a informatiky
GUI	–	Graphical User Interface
GWT	–	Google Web Toolkit
IDE	–	Integrated Development Environment - Vývojové prostředí
IT	–	Informační technologie
JAVA EE	–	Java Platform, Enterprise Edition
JAVA SE	–	Java Platform, Standard Edition
JS	–	JavaScript
MES	–	Manufacturing Execution Systems
OS	–	Operating System - Operační systém
PC	–	Personal Computer - Osobní počítač
PL/SQL	–	Procedural Language/Structured Query Language
RIA	–	Rich Internet application
SJV	–	Středojemná válcovna - závod firmy ArcelorMittal a.s.
SQL	–	Structured Query Language
UI	–	User Interface

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Fakulta elektrotechniky a informatiky nabízí studentům třetího ročníku bakalářského studia dvě možnosti vykonání bakalářské práce. První možností je vypracování bakalářské práce se zvoleným tématem, druhou je vykonání odborné praxe ve firmě, která spolupracuje s FEI, spolu s vypracováním dokumentu o jejím absolvování.

Můj výběr druhé možnosti byl podmíněn zejména tím, že při přijímacím řízení firmy, kromě patřičného vzdělání, vyžadují i určitou praxi, kterou mi škola obecně nemůže v dostačující míře poskytnout. Právě vykonávání individuální odborné praxe mě, mimo jiné, naučilo pracovat v týmu, organizovat si svůj čas a poskytlo náhled „pod pokličku“ života programátora.

Studenti mající zájem o vykonávání odborné praxe si volili firmy v systému KATIS. Zde byl seznam jednotlivých firem s popisem pozic a pracovních náplní. Před nástupem na odbornou praxi jsem již několik let brigádně pracoval na slévárně jako slévarenský dělník, a proto byla firma ATACO spol. s r.o., zaměřující se na vytváření a správu systémů hutních podniků, mou jasnou volbou.

V úvodu této práce blíže popíši firmu ATACO spol. s r.o. Uvedu technologie využívané při mé praxi, jako například Java, Vaadin a další. Zmíním jednotlivé úkoly, s nimiž jsem se v rámci praxe potýkal, spolu s popisem jejich řešení. Dále přiblížím dovednosti, které jsem při vykonávání praxe získal a vyzdvihnu ty teoretické a praktické znalosti získané v průběhu studia, jenž byly nezbytné při plnění pracovních úkolů. Závěrem se zamyslím nad přínosem praxe pro můj profesní rozvoj.

2 Popis odborného zaměření firmy a pracovní zařazení studenta

2.1 O firmě ATACO spol. s r.o.

Česká firma ATACO[1] spol. s r.o., sídlící v Podnikatelském inovačním centru (BIC) v Ostravě Vítkovicích na ulici Ruská 83/24, založena v roce 1991 je dodavatelem technologií, řešení a IT služeb pro podnikovou sféru. Její specializací je zejména poskytování komplexních dodávek zakázkového software pro velké průmyslové podniky. Mezi další služby patří vývoj krabicového software pro malé a střední podnikání, konzultační a analytická činnost. Společnost zajišťuje komplexní dodávky výpočetní techniky a síťových prvků včetně dodávek špičkové techniky pro bezkontaktní měření.

Oblasti činnosti firmy:

- Hutnictví – zejména řízení technologických a výrobních procesů a řízení údržby.
- Klimatologie – díky spolupráci s Českým hydrometeorologickým ústavem je autorem celosvětově užívané klimatologické aplikace CLIDATA
- Malé a střední podnikání – dodavatel aplikací podporujících rozvoj a efektivitu
- Internetové služby – zejména vývoj a implementace webových aplikací

Logo ATACO spol. s r.o.

2.2 Popis pracovního zařazení studenta

Ve firmě jsem pracoval jako junior JAVA programátor. Hlavní pracovní náplní bylo vytvoření webové aplikace pro zobrazování, editaci a klasifikaci prodlev vznikajících na nůžkách ve Středojemné válcovně (SJV). Tuto aplikaci jsem vytvářel v jazyce JAVA ve frameworku Vaadin, jako aplikační server byl zvolen WildFly a pro zabezpečení aplikace byl využit KeyCloak.

3 Seznam úkolů zadaných studentovi

3.1 Popis zadaných úkolů

3.1.1 Instalace softwaru a seznámení s projektem

První dny byly věnovány instalaci veškerého potřebného softwaru a také podrobným seznamováním se s projektem, který budu realizovat.

3.1.2 Vytvoření databázových tabulek

Bylo potřeba vytvořit dvě tabulky v databázi SJV. Jedna obsahovala kódy prodlev s jejich popisy, druhá samotné prodlevy.

3.1.3 Vytvoření modulu pro automatické zaznamenávání prodlev do databáze

Dle požadavků zákazníka měly vznikat prodlevy vždy, pokud nůžky neprovedly po určitou dobu střih. Bylo tedy nutné vytvořit modul, který bude vytvářet nový záznam v databázi, pokud po určitou dobu neobdrží telegram značící provedení střihu.

3.1.4 Verzovaní pomocí GIT klienta

Při vývoji webové aplikace byl využit verzovací nástroj GIT. Mým úkolem bylo propojit IDE s GIT repositářem a naučit se s ním pracovat.

3.1.5 Vytvoření lokální databáze

Cílem bylo vytvoření lokální Oracle databáze, která měla sloužit jako testovací.

3.1.6 Vytvoření webové aplikace

Bylo potřeba vytvořit webovou aplikaci pro zobrazení, klasifikaci a úpravu vzniklých prodlev. Dalším požadavkem bylo vytvořit GUI podle vize, kterou jsem dostal nakreslenou na papíře spolu s aplikační logikou. To jsem realizoval vytvořil ve frameworku Vaadin, který firma již několik let používá. Samotná webová aplikace měla běžet na aplikačním serveru WildFly.

3.1.7 Autentizace uživatelů s užitím KeyCloak

Dále bylo potřeba vytvořit uživatelské role a aplikaci zabezpečit. Tuto úlohu jsem realizoval nástrojem zvaný KeyCloak.

3.1.8 Vytvoření uživatelské dokumentace

Uživatelská dokumentace měla být vytvořena v Open Office a to tak, aby po jejím přečtení mohl tuto aplikaci obsluhovat každý, bez ohledu na zručnost a zkušenosti s prací na PC.

3.1.9 Vytvoření programátorské dokumentace

Mezi mé úkoly patřilo také řádně zdokumentovat veškeré třídy a metody dle standardu Javadoc.

3.1.10 Nasazení a předvedení aplikace zákazníkovi

Hotovou aplikaci bylo potřeba nasadit na ostrý server. Poté byla domluvena schůzka se zaměstnanci firmy, kterým jsem měl aplikaci předvést.

3.2 Časová náročnost zadaných úkolů

Úloha	Počet dní
Instalace softwaru a seznámení s projektem	2
Vytvoření databázových tabulek	1
Vytvoření modulu pro automatické zaznamenávání prodlev do databáze	9
Verzovaní pomocí GIT klienta	2
Vytvoření lokální databáze	2
Vytvoření webové aplikace	34
Autentizace uživatelů s užitím KeyCloak	5
Vytvoření uživatelské dokumentace	3
Vytvoření programátorské dokumentace	1
Nasazení a předvedení aplikace zákazníkovi	1

3.3 Technologie potřebné pro plnění zadaných úkolů

3.3.1 Java Enterprise Edition

Java Enterprise Edition[2] je založena na Java SE. Poskytuje platformu pro vývoj webových a podnikových aplikací, pro přístup k relačním databázím, přístup k legacy systémům a také podporuje vývoj sdílené business logiky – Enterprise Java Beans (EJB). Aplikace, pro platformu Java EE, se vyvíjejí na základě API a dalších definovaných fragmentů dle jednotlivých specifikací. Takto vyvinuté aplikace běží na aplikačním serveru.

3.3.2 WildFly

Jako aplikační server byl využit WildFly[5], vyvíjen firmou Red Hat. Hlavní výhodou je jeho open-source licence a využití GNU LGPL s podporou množství platforem(Microsoft Windows, Mac OS X, Linux).

3.3.3 Vaadin

Framework Vaadin[3] se využívá pro tvorbu RIA webových aplikací. Výhodou je, že vývojář píše aplikaci v Javě a nemusí, ani by neměl, zasahovat do výsledného kódu, který je přeložen GWT do JS a následně se zobrazuje pomocí prohlížeče. GUI se ve Vaadinu píše velice podobně jako GUI aplikace desktopové.

3.3.4 GIT

GIT[4] je systém pro správu verzí. Pro mou aplikaci mi byl přidělen GIT repozitář, kde byla uložena vždy aktuální verze aplikace.

3.3.5 JUnit

Unit testování probíhalo pomocí frameworku JUnit[6]. Jedná se o nejrozšířenější framework pro testování Java aplikací. Testuje se zpravidla jedna metoda, případně třída. Pomocí JUnit lze testovat jak logika kódu, tak integrita mezi komponenty.

3.3.6 KeyCloak

Pro správu přístupu do aplikace byl využit KeyCloak[7], což je opensource vyvinut firmou Red Hat. Obsahuje management pro správu účtů a rolí. KeyCloak lze jednoduše integrovat do aplikace bez nutnosti složitě měnit její kód. Umožňuje také SSO, zajišťující přihlášení do množství aplikací pomocí jediného přihlášení do KeyCloaku.

3.3.7 Oracle Database

Databázový systém firmy Oracle[8] poskytuje široké možnosti zpracování dat formou relačních databází. Systém může fungovat na množství platforem. Pro práci s daty lze kromě klasického dotazovacího jazyka SQL využít imperativní programovací jazyk PL/SQL umožňující, mimo jiné, tvorbu vlastních procedur, funkcí i triggerů.

4 Postup při řešení zadaných úkolů

První dny jsem strávil seznamováním se s projektem a požadavky, což probíhalo formou briefingů. Bylo mi také přiděleno místo v kanceláři spolu s notebookem s čistou instalací Microsoft Windows. Firma mi dala naprostou volnost ve volbě jednotlivých programů. Jako IDE jsem si zvolil program Eclipse Neon, podporující řadu programovacích jazyků včetně řady plug-inů.

4.1 Vytvoření databázových tabulek v databázi SJV

Díky mnoha konzultacím jsem měl jasnou představu o tom, jak má databáze vypadat. Měl jsem vytvořit dvě tabulky, kde jedna obsahuje pouze dva sloupce a druhá samotné informace o prodlevách. Pro práci s databází jsem použil program SQuirreL. Přidal jsem potřebný driver pro práci s Oracle databází a přihlásil se k databázi SJV. Samotné tabulky jsem vytvořil pomoci jednoduchého skriptu viz. výpis 1.

```
CREATE TABLE DB.PROSTOJE_KODY
(
K_KOD decimal(3) PRIMARY KEY NOT NULL,
K_POPIS varchar2(255)
);
CREATE UNIQUE INDEX PROSTOJE_KODY_PK
ON DB.PROSTOJE_KODY(K_KOD);

CREATE TABLE DB.PROSTOJE
(
   P_ZACATEK timestamp NOT NULL,
   P_KONEC timestamp,
   P_STRANA char(1) NOT NULL,
   P_KOD decimal(3),
   P_POPIS varchar2(80)
);
ALTER TABLE DB.PROSTOJE
ADD CONSTRAINT P_FK_KOD
FOREIGN KEY (P_KOD)
REFERENCES DB.PROSTOJE_KODY(P_KOD);
```

Výpis 1: Script pro vytvoření databázových tabulek

4.2 Vytvoření modulu pro automatické ukládání vzniklých prodlev

Před samotným vytvořením modulu jsem musel pochopit, jak fungují řidící systémy SJV. Značný čas jsem tedy strávil studiem dokumentací, případně zaškolováním se mými kolegy.

Řidící systémy jsou rozděleny do několika úrovní, tyto úrovně mezi sebou komunikují pomocí komunikačních toků:

- Úroveň L1 - automatizace. Tvoří ji samotné signály o událostech, například o střihu nůžek.
- Úroveň L2 - zpracovává signály z L1 a reaguje na ně. Na základě signálů spravuje záznamy v databázi, případně poskytuje data vyšší úrovni.
- Úroveň L3 - MES. Při práci s daty vrstvy L2 využívá abstraktních informací, například směnový kalendář. Na této vrstvě běží programy pro poskytování různých analýz, informací o výrobě atp.

Modul pro ukládání prodlev tedy měl běžet na serveru nacházejícím se na logické úrovni L2 a musel být napsán v jazyce C++. Samotná aplikace pro zobrazování a editaci prodlev bude pracovat na aplikačním serveru, který patří do úrovně L3.

Metoda, pro zpracování signálu o střihu nůžek již existovala, musel jsem tedy pouze zajistit samotný zápis prodlevy do databáze, pokud daný signál o střihu nůžek nebyl touto metodou zaevidován. Jednalo vlastně o periodicky se opakující cyklus, který po uplynutí daného času zkontroloval, zda proběhl střih.

```
while(1){
   kontrola = receive_notify("TIMER", &value, timeout);

   if (kontrola==-1) {
      //vytvorit prostoj
      EXEC SQL INSERT INTO DB.PROSTOJE
      (ZACATEK,STRANA)
      values
      (sysdate-:timeout/86400,'L');

      if (sqlca.sqlcode!=0){
         printf("\nNalezena chyba Oracle :");
         printf("\n% .70s", sqlca.sqlerrm.sqlerrmc);
         zapis_ora(330,sqlca.sqlcode,sqlca.sqlerrm.sqlerrmc);
         zapis_log(330);//chyba insert into PROSTOJE
      };

      EXEC SQL commit;
```

```
      if (sqlca.sqlcode!=0){
         printf("\nNalezena chyba Oracle :");
         printf("\n% .70s", sqlca.sqlerrm.sqlerrmc);
         zapis_ora(331,sqlca.sqlcode,sqlca.sqlerrm.sqlerrmc);
         zapis_log(331);//chyba commit insert into PROSTOJE
      };

      //dalsi timeout nebude
      timeout = 0;
   } else {
      //ukoncit prostoj
      EXEC SQL UPDATE DB.PROSTOJE set
      KONEC=sysdate
      where
      KONEC is NULL
      and
      STRANA='L';
      if (sqlca.sqlcode!=0 && sqlca.sqlcode!=1403){
         printf("\nNalezena chyba Oracle :");
         printf("\n% .70s", sqlca.sqlerrm.sqlerrmc);
         zapis_ora(332,sqlca.sqlcode,sqlca.sqlerrm.sqlerrmc);
         zapis_log(332);//chyba insert into PROSTOJE
      };

      EXEC SQL commit;
      if (sqlca.sqlcode!=0) {
         printf("\nNalezena chyba Oracle :");
         printf("\n% .70s", sqlca.sqlerrm.sqlerrmc);
         zapis_ora(333,sqlca.sqlcode,sqlca.sqlerrm.sqlerrmc);
         zapis_log(333);//chyba commit update PROSTOJE
      };

      //dalsi timeout
      timeout = 240;
   }
};
```

Výpis 2: Ukázka modulu pro automatické ukládání prodlev

4.3 Propojení vznikajícího projektu s Git repozitářem

Nově vznikající aplikace měla zřízen Git repozitář. Pro zpřístupnění příslušného repozitáře jsem vygeneroval SSH klíč a jeho veřejnou část odeslal správci serveru. Následovalo nastavení Gitu v IDE Eclipse, což jsem realizoval podle návodu staženého z intranetu firmy. Před prvním nahráním na Git bylo nutno nastavit, které složky a soubory mají být na Git nahrány a které má Git ignorovat, ty jsem musel označit ve speciálním souboru `.gitignore`. Následně byl projekt nahrán na Git. Od této chvíle jsem pravidelně aktualizoval veškeré změny aplikace na Git a jelikož jsem na aplikaci pracoval téměř celou dobu sám, nemusel jsem se potýkat s konflikty při operaci Commit a následné operaci Push.

4.4 Vytvoření databáze, doménového modelu a EJB

Bylo mi ukázáno, jak vytvořit databázi na mém lokálním úložišti, v této databázi jsem vytvořil dvě tabulky podle zadání a následně je naplnil několika hodnotami. Pro lepší pochopení problematiky mi spolupracovníci poskytli zdrojový kód již vytvořené podobné aplikace a podle něj jsem pro obě tabulky vytvořil jednoduché DAO a doménový model. V průběhu vývoje aplikace byly vytvořené soubory doplňovány a došlo i k ošetření výjimek, přidání vyhledávacích parametrů a konstant.

```
//Ukazka metody z~DAO, tato metoda vyhleda vsechny zaznamy vyhovujici
    parametrum.
   public final List<T> findAll(Params parameters) {
      Criteria c = getSession().createCriteria(objectClass);
      refineSearch(c, parameters);
      return c.list();
   }
```

Výpis 3: Vyhledání objektů podle parametrů

Po vytvoření pom.xml a nastavení aplikačního serveru WildFly mými kolegy byla má aplikace schopna komunikovat s databází včetně CRUD operací.

4.5 Automatické testy DM

Jelikož jsem se nikdy předtím s testováním tříd a metod pomocí JUnit nesetkal, strávil jsem značnou část řešení tohoto úkolu samostudiem. Po pochopení problematiky byly samotné testy poměrně jednoduché.

```
@Test
/**
* Test vytvoreni tridy DowntimeCode
*/
```

```
public void testCreate() {
   try{
      DowntimeCode tst = new DowntimeCode();
      assertNotNull(tst);
      assertTrue(tst.getCode() == NULL);
      assertTrue(tst.getDescription() == NULL);
      assertTrue(tst.getOldCode() == NULL);
   }catch (Exception e){
      fail(e.getMessage());
   }
}

@Test
/**
* Test kopie, test vsech geteru a~seteru
*/
public void testCreateCopy() {
   DowntimeCode obj = new DowntimeCode();
   obj.setCode(999);
   obj.setOldCode(888);
   obj.setDescription("String");
   assertNotNull(obj);
   DowntimeCode copy = new DowntimeCode();
   copy.setCode(obj.getCode());
   copy.setOldCode(obj.getOldCode());
   copy.setDescription(obj.getDescription());
   assertNotNull(copy);
   assertNotSame(obj, copy);
   assertTrue(copy != obj);
   assertTrue(copy.getCode() == obj.getCode());
   assertTrue(copy.getOldCode() == obj.getOldCode());
   assertTrue(copy.getDescription() == obj.getDescription());
}
```

Výpis 4: Test modelu

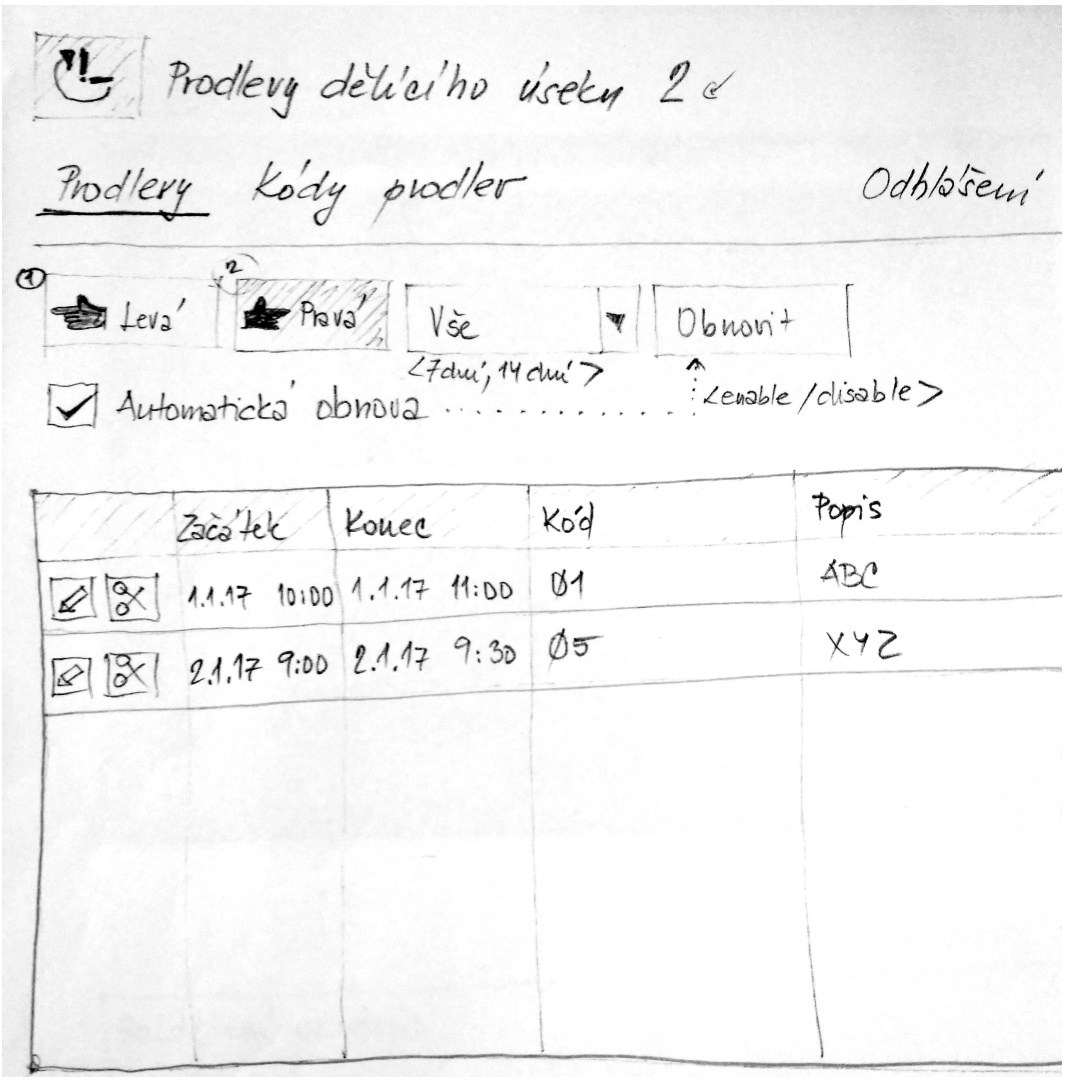
Ve výpisu 4 je znázorněn test třídy DowntimeCode, která má pouze jeden kosntruktor, který nemá žádné parametry. Při vytvoření třídy DowntimeCode očekáváme, že veškeré její proměnné nabývají hodnoty `NULL`. Protože jsou tyto proměnné private, otestoval jsem je pomocí getterů. V testu kopie je vytvořena nová instance třídy, poté jsou proměnné naplněny hodnotami. Poté

je vytvořena nová instance třídy, do které jsou pomocí getterů a setterů překopírována data z první instance. Pokud tyto dvě instance nejsou identické a obsahují stejné hodnoty, pak byl test úspěšný.

Vytvoření podobných testů u ostatních tříd v modelu bylo spíše manuální prací. Díky těmto testům jsem opravil chybu, které bych si nejspíše nevšiml: Třída, očekávající `string` parametr při vytváření instance, se po zadání `NULL` parametru chovala jinak, než jsem očekával.

4.6 Uživatelské rozhraní

Očekávaný vzhled UI jsem dostal na papíře viz. obrázek 1, ten jsem vytvářel prvek po prvku. Základní aplikace obsahovala 2 záložky, a to „Prodlevy“ a „Kódy prodlev“.



Obrázek 1: Návrh karty „prodlevy“

Jednotlivé záložky „Prodlevy“ a „Kódy prodlev“ jsem realizoval pomocí `TabSheet`. Díky čemuž mi Vaadin vytvořil velice elegantní vzhled aplikace.

```
TabSheet tabs = new TabSheet();
tabs.setSizeFull();
tabs.addTab(new DowntimesView());
tabs.addTab(new DowntimeCodesView());
addComponent(tabs);
```

Výpis 5: Vytvoření menu

Filtr tabulek byl vytvořen pomocí `CheckBox`, `ComboBox`, případně `TextField`

```
TextField searchDescription = new TextField(" ");
searchDescription.setNullRepresentation("");
searchDescription.setInputPrompt("Popis");
searchGroup.bind(searchDescription, "description");

Button searchButton = new Button("Vyhledat");
searchButton.addClickListener(new Button.ClickListener() {

@Override
public void buttonClick(ClickEvent event) {
   ErrorHandler.clearErrors(searchGroup);
   try {
      searchGroup.commit();
      DowntimeCodesView.this.onSearchClicked();
   } catch (CommitException ce) {
      ErrorHandler.handleError(ce, searchGroup);
      }
   }
});
```

Výpis 6: Filtr tabulky

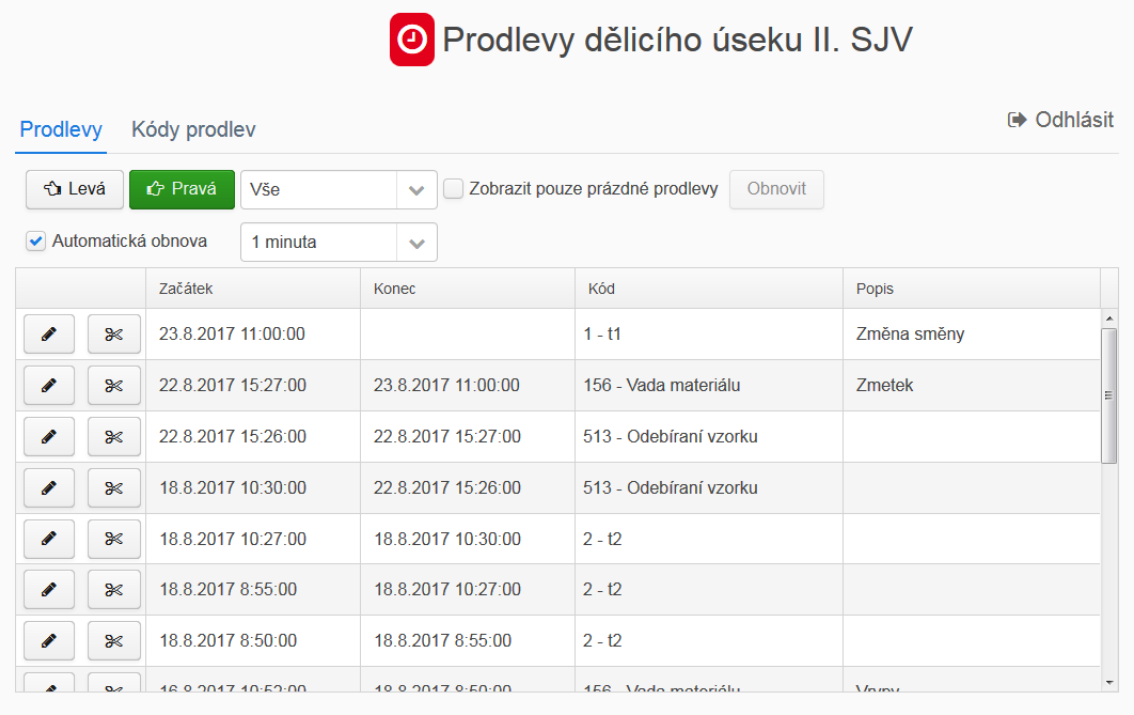
Obrovským problémem se ukázala „tvrdohlavost“ Vaadinu při umísťování prvků do správných pozic. Ladění konečného vzhledu jsem prováděl úpravou stylů, občas jsem si musel vynutit pozici relativním umístěním viz. výpis 7.

```
.signOutButton-style{
   position: relative;
   top: 40px;
```

```
    font-size: 20px;
    color: #696969;
    z-index: 100;
}
.close-button-style{
    margin: 10px;
}
.codes-list-popup-syle{
    padding-top: 10px;
    margin-left: 10px;
}
.v-slot-vertical-error-align{
    vertical-align: top;
}
.vertical-error-align{
    padding-bottom: 3px;
}
```

Výpis 7: Ukázka stylování prvků

Po několika dnech ladění jsem dosáhl kýženého výsledku, ten můžete vidět na obrázku 2.



Obrázek 2: Výsledný vzhled karty „Prodlevy“

4.7 Autentizace uživatelů s užitím KeyCloak s vytvořením uživatelských rolí

Existuje vícero aplikací SJV, které jsou zabezpečeny KeyCloakem a proto bylo potřeba stejným způsobem zabezpečit i tuto nově vzniklou aplikaci.

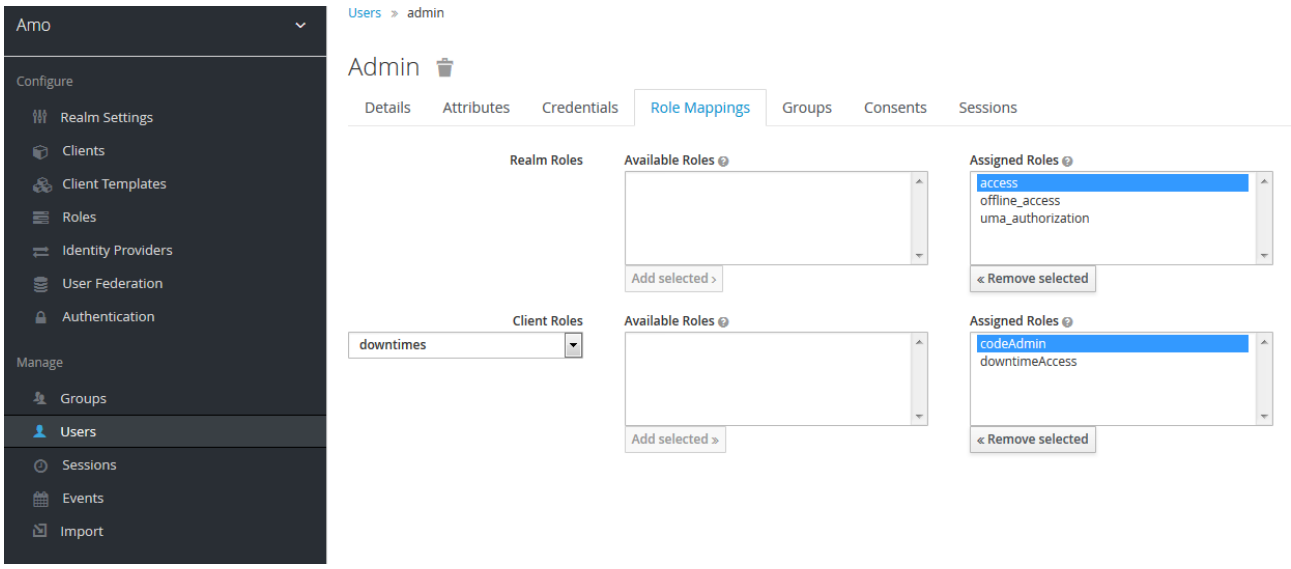
Zabezpečení pomocí KeyCloaku je velice elegantní, postačilo jej pouze nainstalovat na můj počítač a následně pozměnit `web.xml`:

```
<?xml version="1.0"?>
<web-app
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.
    sun.com/xml/ns/javaee/web-app_3_0.xsd"
version="3.0">
<security-constraint>
   <display-name>SecureApplicationConstraint</display-name>
   <web-resource-collection>
      <web-resource-name>Vaadin application</web-resource-name>
      <description>The entire Vaadin application is protected</
          description>
      <url-pattern>/*</url-pattern>
   </web-resource-collection>
   <auth-constraint>
      <role-name>downtimeAccess</role-name>
   </auth-constraint>
</security-constraint>
<login-config>
   <auth-method>KEYCLOAK</auth-method>
   <realm-name>amo</realm-name>
</login-config>
<security-role>
<role-name>downtimeAccess</role-name>
</security-role>
</web-app>
```

Výpis 8: web.xml

Z komunikce se zákazníkem vyvstal požadavek na vytvoření uživatelských rolí, tak, aby editace „Kódů prodlev“ byla možná pouze pro uživatele s určitým oprávněním. Tyto role je možné jednoduše vytvořit přímo ve webovém rozhraní KeyCloaku viz. obrázek 3. Vytvořil jsem tedy roli `downtimeAccess` nutnou pro přístup do aplikace a roli `codeAdmin` nutnou pro možnost

editace „Kódů prodlev“.



Obrázek 3: Přidělení rolí uživatelům ve webovém rozhraní KeyCloaku

Realizace rolí v samotné aplikaci byla vyřešena boolean proměnnou, která nabývala hodnoty `true`, jestliže byl přihlášený uživatel oprávněn v kartě „Kódy prodlev“ provádět změny.

```
tabs.addTab(new DowntimeCodesView(DowntimesUI.getCurrent().
    isUserInRole("codeAdmin")));
```

Výpis 9: Vytvoření záložky kódu prodlev s příznakem

```
if (editMode) {
    boolean isRowEdited = itemId.equals(editingRowId);
    if (isRowEdited) {
        Button save = new Button();
        save.setIcon(FontAwesome.FLOPPY_O);
        save.addClickListener((Button.ClickListener) (event) ->
            DowntimeCodesView.this.onSaveClicked(itemId, event));
        Button cancel = new Button();
        cancel.setIcon(FontAwesome.BAN);
        cancel.addClickListener((Button.ClickListener) (event) ->
            DowntimeCodesView.this.onCancelClicked(itemId, event));
        HorizontalLayout content = new HorizontalLayout(save, cancel);
        content.setSpacing(true);
        return content;
    } else {
        Button edit = new Button();
```

```
            edit.setIcon(FontAwesome.PENCIL);
            boolean enabled = editingRowId == NULL;
            edit.setEnabled(enabled);
            edit.addClickListener((Button.ClickListener) (event) ->
                DowntimeCodesView.this.onEditClicked(itemId, event));
            HorizontalLayout content = new HorizontalLayout(edit);
            content.setSpacing(true);
            return content;
        }
    } else if (window != NULL) {
        Button select = new Button();
        select.setIcon(FontAwesome.CHECK);
        select.addClickListener((Button.ClickListener) (event) ->
            DowntimeCodesView.this.onSelectClicked(itemId, event));
        HorizontalLayout content = new HorizontalLayout(select);
        content.setSpacing(true);
        return content;
    }
```

Výpis 10: Různé zobrazení v závislosti na roli

Tímto byla první verze hotova. Aplikaci jsem předvedl při malém briefingu ve firmě a po drobných úpravách a připomínkách byl domluven datum pro předvedení aplikace zákazníkovi.

4.8 Vytvoření uživatelské dokumentace

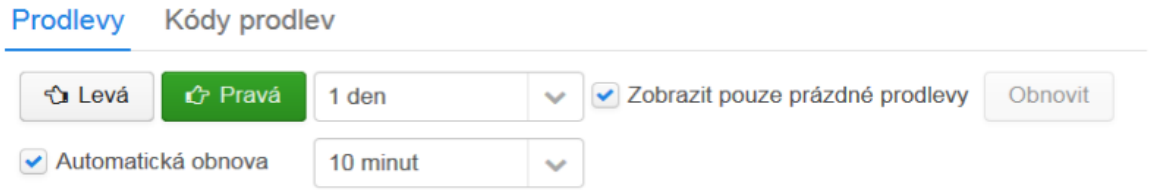
Jako program pro vytvoření uživatelské dokumentace mi byl doporučen OpenOffice, kdy bylo nutné se držet předem připraveného schématu. Při vytváření dokumentace jsem musel popisovat každou, zdánlivě samozřejmou, akci. Psaní dokumentace pro mne bylo příjemným odreagováním, na rozdíl od stresujícího umísťování prvků při tvorbě GUI. Ukázka uživatelské dokumentace je vyobrazena na obrázku 4.

4.9 Nasazení a předání aplikace

Finální verzi aplikace jsem spolu s mým kolegou nahrál na server SJV a důkladně otestoval. Poté jsme jeli hotovou aplikaci prezentovat pracovníkům, kteří budou tuto aplikaci využívat. Ukázal jsem veškeré její funkce, vysvětlil jak funguje a předal uživatelskou dokumentaci.

3 Prodlevy

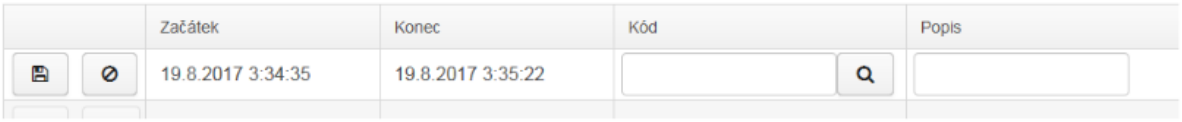
Karta prodlevy zobrazuje údaje o jednotlivých prodlevách a umožňuje tyto prodlevy klasifikovat dle jejich kódů, přidat k nim popis, případně tyto prodlevy rozdělit.



Ilustrace 2: Funkce karty "Prodlevy"

- Tlačítka **Levá** a **Pravá** slouží k přepínaní strany
- Rolovací lištou vyberte, jak staré záznamy si přejete zobrazit
- Zaškrtávací pole "**Zobrazit pouze prázdné prodlevy**" slouží k aplikaci filtru, který zajistí, že se zobrazí pouze takové prodlevy, ke kterým není přiřazen žádný Kód
- Tlačítko **Obnovit** aktualizuje záznamy o prodlevách
- Zaškrtávací pole "**Automatická obnova**" provede aktualizaci tabulky ve zvoleném intervalu. Tento interval můžete změnit v rolovací liště, která se po klepnutí na zaškrtávací pole "Automatická obnova" zobrazí

3.1 Přidání kódu prodlevě



Ilustrace 3: Editace prodlevy

1. U dané prodlevy klikněte na **symbol tužky**
2. V řádku editované prodlevy se zobrazí pole pro zadání **Kódu** a **Popisu**
3. Kliknutím na **symbol lupy** se zobrazí seznam všech kódů, vyhovující kód zvolíte kliknutím na fajfku
4. Pro uložení záznamu klikněte na **symbol diskety.**

Obrázek 4: Ukázka z uživatelské dokumentace

5 Shrnutí použitých, chybějících a získaných dovedností v rámci praxe

Při mé praxi jsem využil řadu dovedností a technik, které jsem získal studiem Vysoké školy báňské – Technické univerzity Ostrava, setkal jsem se ovšem i s množstvím frameworků a programů, jež pro mne byly nové.

5.1 Použité znalosti v rámci praxe získané studiem Vysoké školy báňské – Technické univerzity Ostrava

Základ mé praxe tvořilo programování v jazyce Java, případně v jazyce C++. Pracovat s Javou jsem se naučil v předmětu Programovací jazyky i (PJ1) a také v předmětu Java technologie (JAT), který se také zabýval prací s aplikačním serverem TomCat, jež je velmi blízký aplikačnímu serveru WildFly.

Součástí praxe bylo také vytvoření tabulek a testovací databáze. Využil jsem znalosti z předmětů Úvod do databázových systémů (UDBS) a Databázové a informační systémy (DAIS).

GUI bylo realizováno ve frameworku Vaadin, což se do jisté míry podobalo vytváření GUI pomocí Tkinter v jazyce Python, který jsem znal z předmětu Uživatelská rozhraní (URO). Mimo znalosti Tkinteru mi předmět URO poskytl obecné konvence, jak by mělo GUI vypadat, například seskupování podobných prvků do skupin.

Je třeba podotknout, že použité znalosti není možné takto jednoduše klasifikovat dle určitého předmětu. Zkušenosti jsem získával při studiu postupně, a proto si myslím, že všechny předměty, kde jsem jakkoli programoval a učil se jak aplikace fungují, pro mne byly přínosem.

5.2 Znalosti scházející a získané v průběhu praxe

Na odbornou praxi jsem se přihlásil proto, abych se naučil novým věcem.

Neznámým pro mne byl verzovací systém GIT: Verzovací systémy se při kolektivním vytváření programů a aplikací hojně využívají, zmínit takovéto systémy spolu s jednoduchou praktickou ukázkou při výuce by, podle mého mínění, bylo velice přínosné a vhodné.

Až při praxi jsem se poprvé setkal s frameworkem Vaadin, který má ovšem skvělou dokumentaci a dokonce i jednoduché instruktážní videa, proto mi nedělalo problém jej rychle pochopit.

Jako zásadní nedostatek znalostí získaných studiem hodnotím absenci cvičení zaměřených na JUnit testy. Jedná se o elementární standard, proto pouhá zmínka ve volitelném předmětu v pátém semestru (JAT) nemůže stačit.

6 Závěr

Díky možnosti vykonávat individuální odbornou praxi jsem se naučil množství nových věcí, vytvořil a předal funkční aplikaci, se kterou dnes reálně pracují dělníci. Získané zkušenosti mi pomohou v mém profesním životě. Až praxí jsem pochopil, že programátor je profese, při které se stále učíte něčemu novému, a proto jen tak nezevšední.

Každému studentovi bakalářského programu vysoké školy, s možností vykonání bakalářské praxe namísto tradiční bakalářské práce, vřele doporučuji přemoci prvotní obavy a jít do této možnosti po hlavě. Získat praktické zkušenosti je pro studenta stejně důležité, jako získat titul.

Dominik Sobek

Literatura

[1] *O společnosti ATACO* [Online]. Dostupné z
`http://www.ataco.cz/cs/about/`

[2] *Java EE 7* [Online]. Dostupné z
`http://cdn.oreillystatic.com/oreilly/booksamplers/9781449370176_sampler.pdf`

[3] *Book of Vaadin* [Online] Dostupné z
`https://vaadin.com/download/book-of-vaadin/vaadin-7/pdf/book-of-vaadin.pdf`

[4] *Pro Git*[Online]. Dostupné z
`https://git-scm.com/book/en/v2`

[5] *WildFly 10* [Online]. Dostupné z
`https://docs.jboss.org/author/display/WFLY10/Documentation`

[6] *JUnit 5 User Guide*[Online]. Dostupné z
`http://junit.org/junit5/docs/current/user-guide/`

[7] *KeyCloak*[Online]. Dostupné z
`http://www.keycloak.org/about.html`

[8] *Oracle Database 11g The Complete Reference*[Online]. Dostupné z
`https://anargodjaev.files.wordpress.com/2013/09/`
`oracle-database-11g-the-complete-reference.pdf`